**a. $0** – The name of the command

**b. $1 - $9** – Command line arguments;

$n the n-th argument; shift n to shift cmd line arguments by n

**c. $\* or $@** - All the arguments together as string or as array

**d. $#** - Number of command line arguments

**e. $?** – Exit code of the previous command

Basic example using-for.sh, showing the do on the same line or on the next line. Semicolon is the command separator. a. chmod 700 using-for.sh b. ./using-for.sh

#!/bin/bash

for A in a b c d; do

echo Here is $A

done

for A in a b c d

do

echo Here is $A

done

**test expression**  
-------------------  
**-d file** = True if the file exists and is a directory.  
**-f file** = True if the ile exists and is a regular file  
**-p file** = True if the file exists and is a named pipe.  
**-r file** = True if the file exists and is readable.  
**-w file** = True if the file exists and is writable.  
**-x file** = True if the file exists and is executable.  
**-z string** = True if the length of the string is 0.  
**-n string** = True if the length of the string is non-zero.  
**string1 = string2** = True if the strings are equal.  
**string1 != string2** = True if the strings are not equal.  
**!expr** = True if the expr evaluates to false.  
e**xpr1 –a expr2** = True if both expr1 and expr2 are true.  
**expr1 –o expr2** = True is either expr1 or expr2 is true.

1. Write a bash script that calculates the sum of the sizes (in bytes) of all regular files in a folder given as a parameter.(use test to check if the folder exists and if a given file is a regular file)

#!/bin/bash

if [ -z "$1" ]; then

echo "No parameters given"

exit 1

fi

if [ ! -d "$1" ]; then

echo "Parameter is not a folder"

exit 1

fi

sum=0

for item in $(ls "$1");do

file="$1/$item"

if [ -f "$file" ]; then

size=`ls -l $file | awk '{print $5}'`

echo Size of file $item is $size

sum=$((sum+size))

else

echo Not a file

fi

done

2. Write a script that reads filenames until the word "stop" is entered. For each filename, check if it is a text file and if it is, print the number of words on the first line.(Hint: test command to check if regular file; file command to check if text file)

#!/bin/bash

while true; do

read filename

if [ "$filename" == "stop" ]; then

break

else

if [ -f $filename ];then

n=`file $filename`

# check if it is a .txt or not

if [ "$n" == "$filename: ASCII text" ]; then

echo "It is a text file"

# get the count of words on the first line

words=`cat $filename | head -n 1 | wc -w`

echo "$filename is a file and the word count on first line is $words"

fi

else

echo $filename it is not a file

fi

fi

done

3. Write a script that receives as command line arguments pairs consisting of a filename and a word. For each pair, check if the given word appears at least 3 times in the file and print a corresponding message.

#!/bin/bash

if [ $# -lt 2 ]; then

echo "Please provide at least 2 arguments"

exit 1

fi

if [ $(($# % 2)) -eq 1 ]; then

echo "You must provide an even number of arguments"

exit 1

fi

while [ $# -gt 1 ]; do

file=$1

word=$2

if [ ! -f "$file" ]; then

echo "Name $file is not a file"

else

count=$(grep -E -o "\<$word\>" "$file" | wc -l)

if [ $count -ge 3 ]; then

echo "Word $word appears $count times in file $file"

fi

fi

shift 2

done

if [ $# -eq 1 ]; then

echo "Warning: final pair is incomplete"

fi

4. Write a bash script that sorts all files given as command line arguments descending by size.(first check if an argument is a file)

#!/bin/bash

declare -A dimensiuni

for X in $@; do

if [ -f $X ]; then

N=`ls -l $X | awk '{print $5}' # ii iau dimensiunea fisierului`

dimensiuni[$X]=$N

fi

done

for key in ${!dimensiuni[@]}; do

echo $key '-' ${dimensiuni["$key"]}

done | sort -rn -k3 #sorteaza outputul forului dupa al 3 lea camp ( dimensiunea)

5. Write a script that extracts from all the C source files given as command line arguments the included libraries and saves them in a file.(use the file command to check if a file is a C source file)

#!/bin/bash

# un script care extrage bibliotecile din fisiere C si le salveaza in alt fisier

for X in $@; do # iterez prin argumente

if [ ${X: -2} == ".c" ]; then # verifica daca ultimile 2 caractere sunt ".c"

echo `grep -E "^(#include).\*" $X | awk '{print $2}'` >> biblioteci.txt # presupun ca e #include(spatiu)<biblioteca>

echo

fi

done

7. Find recursively in a given directory all the symbolic links, and report those that point to files/directories that no longer exist. Use option -L to test if a path is a symbolic link, and option -e to test if it exists (will return false if the target to which the link points does not exist)

#!/bin/bash

for link in $(find "$1" -type l); do

if [ ! -e "$link" ]; then

echo "Link $link is not valid"

fi

done

8. Write a bash script that receives a folder name as argument. Find recursively in the folder the number of times each file name is repeated.  
Scrieti un script bash care primeste un nume de director ca parametru. Cautati recursiv in director si numarati aparitiile fiecarui nume de fisier.

#!/bin/bash

if [ -z "$1" ]; then

echo "Please provide one argument"

exit 1

fi

if [ ! -d "$1" ]; then

echo "Argument must be a directory"

exit 1

fi

find "$1" -type f | awk -F/ '{print $NF}' | sort | uniq -c

11. Write a script that receives a directory as a command line argument. The script will delete all the C source files from the directory and will display all other text files sorted alphabetically.

#!/bin/bash

# primeste un director si sterge toate fisierle .c si afiseaza celelallte fisiere sortate

if [ ! -d $1 ]; then

echo "Nu mi ai dat un director"

exit 1

fi

for F in `find $1 -type f`;do

Tip=`file $F`

# echo $F '-' $Tip

if [[ "$Tip" == \*"C source"\* ]]; then # exit codul lui grep va fi 1 daca gaseste C source

`rm $F`

echo "Am sters fisierul $F"

else

echo $F >> notC.txt # appenduiesc numele fisireului in unul text dupa care il sortez si afisez

fi

done

echo `cat notC.txt | sort | less `

`rm notC.txt` # fac curatenie dupa mine

12. Write a script that finds recursively in the current folder and displays all the regular files that have write permisions for everybody (owner, group, other). Then the script removes the write permissions from everybody. Hint: use chmod's symbolic permissions mode (see the manual).

RWX="??w??w??w?"

for X in `find . -type f`; do

Perm=`ls -l $X | awk '{print $1}'`

#-rwxrwxrwx

if [[ $Perm =~ ..w..w..w. ]];then #am folosit =~ expresie regulara fiindca nu a mers cu wildcardul ??w??w??w?

echo $X "Are drepturi de citire pentru toti"

`chmod u-w,g-w,o-w $X`

fi

done

13. Consider a file containing a username on each line. Generate a comma-separated string with email addresses of the users that exist. The email address will be obtained by appending "@scs.ubbcluj.ro" at the end of each username. Make sure the generated string does NOT end in a comma.

#!/bin/bash

if [ -z "$1" ]; then

echo "Please provide one input file"

exit 1

fi

if [ ! -f "$1" ]; then

echo "The given argument is not a file"

exit 1

fi

result=""

for u in $(cat "$1"); do

result="$u@scs.ubbcluj.ro,$result"

done

result=$(echo $result | sed -E "s/,$//")

echo $result

14. Write a shell script that recieves any number of words as command line arguments, and continuously reads from the keyboard one file name at a time. The program ends when all words received as parameters have been found at least once across the given files.

Example:

Let say that

file1.txt contains word1 and word2

file2.txt does not contain any of the 3 words

file3.txt contains word2 and word 3

./script.sh word1 word2 word3

We input the following:

file1.txt

file2.txt

file3.txt

The program stops after reading file3.txt because

word1 has been found in file1.txt

word2 has been found in file1.txt and file3.txt

word3 has been found in file3.txt

#!/bin/bash

declare -A words

for i in $@; do

words[$i]=0

done

found\_all=false

while ! $found\_all; do

found\_all=true

read -p "Input a file name: " file

if [ -z "$file" ]; then

echo "Empty input"

elif [ ! -f "$file" ]; then

echo "Not a file"

else

for word in ${!words[@]}; do

if grep -q -E "\<$word\>" "$file"; then

echo "Found $word in $file"

words[$word]=1

fi

if [ 0 -eq ${words[$word]} ]; then

found\_all=false

fi

done

fi

done

echo "All done"

15. Write a shell script that, for all the users in /etc/passwd, creates a file with the same name as the username and writes in it all the ip addresses from which that user has logged in. (hint: use the last command to find the ip addresses)

#!/bin/bash

for X in `cat ../passwd | awk -F: '{print $1}'`; do

Ip=`cat ../last.fake | grep -E "$X"`

echo $Ip > $X.txt

done

1. Display a report showing the full name of all the users currently connected, and the number of processes belonging to each of them.

#!/bin/bash

users=`cat materiale/who.fake | awk '{print $1}' | sort | uniq`

echo $users

for elem in $users

do

cat materiale/ps.fake | awk -v user=$elem 'BEGIN{nr=0}{if($1==user)nr=nr+1;}END{print user " " nr}'

done

2. Find recursively in a directory all ".c" files having more than 500 lines. Stop after finding 2 such files.

#!/bin/bash

c\_files=`find materiale/dir -type f -name "\*.c"`

nr=0

for elem in $c\_files

do

nr\_linii=`wc -l $elem | awk 'END{print $1}'`

if [ $nr\_linii -gt "500" ]

then

echo $elem

nr=`expr $nr + 1`

if [ $nr -ge "2" ]

then

break

fi

fi

done

3. Find recursively in a directory, all the files with the extension ".log" and sort their lines (replace the original file with the sorted content).

#!/bin/bash

files=`find materiale/dir -type f -name '\*.log'`

for elem in $files

do

cat $elem | sort > aux.txt

cat aux.txt > $elem

done

4. Find recursively in a given directory all the symbolic links, and report those that point to files/directories that no longer exist. Use operator -L to test if a path is a symbolic link, and operator -e to test if it exists (will return false if the target to which the link points does not exist)

#!/bin/bash

files=`find materiale/dir -type l`

for elem in $files

do

if [ ! -e $elem ]

then

echo $elem

fi

done

5. Write a script that receives dangerous program names as command line arguments. The script will monitor all the processes in the system, and whenever a program known to be dangerous is run, the script will kill it and display a message.

#!/bin/bash

while [ true ]

do

for elem in $@

do

echo "E timpul sa verific daca nu s-a strecurat vreun proces problematic!"

procese=`ps -ef |grep -E -v "$0" |awk -v vir=$elem '{if($NF == vir){print $2}}'`

for proces in $procese

do

kill -9 $proces

echo "Aha! Te-am gasit, proces problematic cu PID $proces!"

done

done

sleep 10

done

6. Find recursively in a directory, all the files that have write permissions for everyone. Display their names, and the permissions before and after removing the write permission for everybody. You will need to use chmod's symbolic permissions mode, instead of the octal mode we have used in class. The the chmod manual for details.

#!/bin/bash

fisiere=`find $1 -type f`

for elem in $fisiere

do

rez=`ls -l $elem | awk '{if($1 ~ "w.$")print $1 " " $9}'`

if [ ! -z "$rez" ]

then

echo $rez

chmod o-w $elem

ls -l $elem | awk '{print $1 " " $9}'

fi

done

8. Display all the mounted file systems who are either smaller than than 1GB or have less than 20% free space.

cat materiale/df.fake | sed -E "y/M%/ /" | awk '{if(NR!=1){if(($2<1024)||($5 > 80)){print $6}}}'

9. Write a script that finds in a given directory hierarchy, all duplicate files and displays their paths. Hint: use checksums to detect whether two files are identical.

#!/bin/bash

fisiere=`find $1 -type f`

if [ -f aux.txt ]

then

rm aux.txt

fi

for fisier in $fisiere

do

var=`md5sum $fisier| awk '{print $1}'`

echo $fisier | awk -F\/ -v cod=$var '{print $NF " " cod " " $0}' >> aux.txt

done

cat aux.txt | awk '{print $1 " " $2}' | sort | uniq -c | awk '{if($1>1){print $2 " " $3}}' | while read linie

do

grep -E "^$linie" aux.txt | awk '{print $3}'

done

10. Display the session count and full names of all the users who logged into the system this month, sorting the output by the session count in descending order. Use the -s and/or -t options of command last to get this month's sessions, and the command date to generate the required timestamp in the expected format.

#!/bin/bash

luna=`date| awk '{print $2}'`

last | awk -v l=$luna '{if($5==l){print $1}}' | sort | uniq -c | sort -nr | while read linie

do

de\_afisat1=`echo $linie | awk '{print $1}'`

var=`echo $linie | awk '{print $2}'`

de\_afisat2=`grep -E "^$var:" /etc/passwd | awk -F: '{print $5}'`

de\_afisat="$de\_afisat1 $de\_afisat2"

echo $de\_afisat

done

1. Write a shell script that reads a number N1 and then reads filenames, ignoring input that is not a file name, until the string “stop” is entered. For each filename the script will check if it contains N1 number of lines, and if it is so, it adds the names of those files to a file called solt.txt. At the end print the contents of sol.txt

#!/bin/bash

read n

while true;do

read filename

if [ "$filename" == "stop" ]; then

break

else

if [ -f $filename ]; then

nrLines=`wc -l $filename | awk 'END{print $1}'`

if [ $nrLines -eq $n ]; then

echo $filename > sol.txt

fi

fi

fi

done

2. Write a shell script that receives as command line arguments pairs of the form file1 number1. The script checks if the file file1 contains on the first line number1 words, and if it does, it adds file1 to a file called sol2.txt, sorting them alphabetically. This is performed for each pair. Display the contents of sol.txt at the end. Ignore filenames that do not exist

#!/bin/bash

if [ $# -lt 2 ]; then

echo "Please provide at least 2 arguments"

exit 1

fi

if [ $(($# % 2)) -eq 1 ]; then

echo "You must provide an even number of arguments"

exit 1

fi

while [ $# -gt 1 ]; do

file=$1

n=$2

if [ -f "$file" ]; then

countWords=`cat $file | head -n 1 | wc -w`

if [ $countWords == $n ]; then

echo $file > sol2.txt

fi

fi

shift 2

done

echo ` cat sol2.txt | sort | less `

`rm sol2.txt`

**Write a shell script that receives any number of command line arguments of the type usernames, files, and directories, and processes each argument such that:**

* **For each username, print the full name of the user. If it’s logged in, print the number of running processes. If it’s not logged in, print the start login time, IP and duration of the last login session. Check if the username exists**
* **For the files that have both read and write permissions, replace all negative numbers (integers or not) in the file with “ERROR”**
* **For each directory print its size in bytes**.

#!/bin/bash

2

3 if [ $# -lt 1 ]; then

4 echo "Please provide at least 1 argument"

5 exit 1

6 fi

7

8 users=`who | awk '{print $1}' | sort | uniq` #get list of users

9

10 for X in $@; do

11 if [ -d $X ]; then

12 # size=`ls -l sed | awk '{s=s+($5)}END{print s}'`

13 size=`du -s -B1 $X | awk '{print $1}'`

14 echo $X is a directory with $size bytes

15 elif [ -f $X ]; then

16 echo $X is a file

17 if [ -r $X ] && [ -w $X ]; then

18 sed -i -E 's/-[0-9]+\.\*[0-9]\*/ERROR/g' $X

19 fi

20 elif id "$X" &>/dev/null; then #if user exists

21 name=`grep $X /etc/passwd | cut -d: -f5`

22 echo $X is a username with the name $name

23 status=0

24 for u in $users; do

25 if [ "$u" == "$X" ]; then

26 # user is logged

27 status=1

28 fi

29 done

30 if [ $status == 1 ]; then

31 number=`ps -U $X | head -n -1 | wc -l`

32 echo "User is logged in with $number processes "

33 else

34 lastTime=`last $X | head -n 1 | awk '{print "IP is " $3", start login time is "$7", duration is "$10}'`

35 echo "User is not logged in. $lastTime"

36 fi

37 else

38 echo "User $X not found"

39 fi

40 done

41

**Write a shell script that receives any number of command line arguments of the type usernames, files, and directories, and processes each argument such that:**

* **For each username, print the full name of the user. If it’s logged in, print the number of running processes. If it’s not logged in, print the start login time, IP and duration of the last login session. Check if the username exists**
* **For the files that have both read and write permissions, replace all positive numbers (integers or not) in the file with “ERROR”**
* **For each directory print its permissions.**

1 #!/bin/bash

2

3 if [ $# -lt 1 ]; then

4 echo "Please provide at least 1 argument"

5 exit 1

6 fi

7

8 users=`who | awk '{print $1}' | sort | uniq`

9

10 for X in $@; do

11 if [ -d $X ]; then

12 permissions=`ls -ld $X | awk '{print $1}'`

13 echo $X is a directory with permissions $permissions

14 elif [ -f $X ]; then

15 echo $X is a file

16 if [ -r $X ] && [ -w $X ]; then

17 sed -i -E 's/(^| )[0-9]+\.\*[0-9]\*/ ERROR/g' $X

18 fi

19 elif id "$X" &>/dev/null; then

20 name=`grep $X /etc/passwd | cut -d: -f5`

21 echo $X is a username with the name $name

22 status=0

23 for u in $users; do

24 if [ "$u" == "$X" ]; then

25 # user is logged

26 status=1

27 fi

28 done

29 if [ $status == 1 ]; then

30 number=`ps -U $X | head -n -1 | wc -l`

31 echo "User is logged in with $number processes "

32 else

33 lastTime=`last $X | head -n 1 | awk '{print "IP is " $3", start login time is "$7", duration is "$10}'`

34 echo "User is not logged in. $lastTime"

35 fi

36 else

37 echo "User $X not found"

38 fi

39 done

**Write a shell script that receives as command line arguments a character sequence without whitespace followed by any number of arguments. For each argument:**

**-if the argument is a regular file, the script will report if the file contains that sequence**

**-if the argument is a folder, the script will report if it contains an item with the name equal to the character sequence(where item can be a file, folder, link , etc)**

**-else, the script will ignore the argument**

**The script will perform all necessary checks**

1 #!/bin/bash

2

3 if [ $# -le 1 ]; then

4 echo "Please give at least 2arguments"

5 exit 1

6 fi

7

8 seq=$1

9 shift

10

11 for x in $@;do

12 if [ -f $x ]; then

13 if grep -E -q "$seq" "$x"; then

14 echo "File $x contains sequence: $seq"

15 else

16 echo "File $x doesn't contain sequence: $seq"

17 fi

18 elif [ -d $x ]; then

19 if ls $x | grep -E -q "^$seq$"; then

20 echo "Folder $x contains item: $seq"

21 elif ls $x | grep -E -q "/$seq$"; then

22 echo "Folder $x contains item: $seq"

23 fi

24 fi

25

26 done

**Create a shell script that receives as a command line argument a folder name. The script will search recursively in the folder and will display:**

**-the names of the files that do not have the “.txt” extension**

**-the contents of the file that do have the”.txt” extension but replacing lowercase vowels with uppercase vowels**

**The script should perform all the necessary checks**

1 #!/bin/bash

2

3 if [ $# -lt 1 ]; then

4 echo "Please give at least 1 argument"

5 exit 1

6 fi

7

8 d=$1

9

10 for f in $(find "$1" -type f); do

11 if [[ $f == \*.txt ]]; then

12 echo $f is a .txt file and it contains:

13 sed 'y/aeiou/AEIOU/' $f

14 else

15 echo $f does not have a .txt extension

16 fi

17 done

-----------------------------------------------------

echo "Numarul total de linii de cod din toate fisierele sursa care nu au extensia .c"

for f in $(find "$1" -type f | grep -E -v "\.c"); do

if file $f | grep -E -q "C program"; then

echo $f

fi

done

echo "Numele fisierelor de tip .txt"

for f in $(find "$1" -type f | grep -E "\.txt"); do

echo $f

done

echo "Numele fisierelor sursa sh care au extensia .sh:"

for f in $(find "$1" -type f | grep -E "\.sh"); do

#echo `file $f`

if file $f | grep -E -q "shell script"; then

echo $f

fi

done

------------------------------------------------------------

**Sa se scrie un script shell care: va primi ca argumente un sir reprezentand drepturi de acces (de forma : rw-) urmat de oricate nume de directoare. Pt fiecare director:**

**-daca directorul nu exista, scriptul va afisa numele directorului si ca nu exista**

**-daca exita:**

**-pt fiecare fisier al carui proprietar are drepturile de acces date, scriptul va crea o copie de siguranta cu acelasi nume si extensia .bak**

**-drepturile de acces ale copiei vor fi modificate astfel incat numai proprietarul fisierului sa aiba dreptul de a citi fisierul**

**-pentru fiecare fisier cu drepturile de acces specificate, scriptul va afisa urm informatii:**

**Permisiuni: rw-**

**Director : /home/../test**

**Fisieer: /home/../test/file1**

**Permisiuni: -rw-r—r—**

**Copie: /home/../test/file1.bak**

**Permisiuni: -r-------**

**Director: ....**

#!/bin/bash

2

3 if [ $# -lt 2 ]; then

4 echo "Numar insuficient de argumente"

5 exit 1

6 fi

7

8 for D in $@; do

9 if [ "$D" == "$1" ]; then

10 continue

11 fi

12 if [ ! -d $D ]; then

13 echo Director: $D

14 echo Directorul dat nu exista.

15 else

16 echo Permisiuni: $1

17 for f in $(find "$D" -type f); do

18 permission=`ls -ld $f | cut -d ' ' -f1 | cut -c 2-4`

19 if [ "$permission" == "$1" ]; then

20 echo Director: $D

21 echo " Fisier: $f"

22 echo " Permisiuni: `ls -l $f | cut -d ' ' -f1` "

23 `cp $f $f.bak`

24 `chmod go-r $f.bak`

25 echo " Copie: $f.bak"

26 echo " Permisiuni: `ls -l $f.bak | cut -d ' ' -f1` "

27 `rm $f.bak`

28 fi

29 done

30 fi

31 done

**Create a shell script that will receive as command line arguments a list of possible files or directories. The script will search recursively in the current directory and will do the following:**

**-if the argument indicates the name of an existing file, it will store in a file named file-reports.info the full path of the file and its size**

**-if the argument indicates the name of an existing directory, it will store in a file named directory-reports.info the name of the largest file in that directory and its size (no recursion)**

**-if it was not found a file/directory with the value of the argument, it will be created a file if the value starts with a vowel, respectively a directory if the value does not start with a vowel.**

**The created file/directory will be named with the value of the argument**

#!/bin/bash

if [ -z $1 ]; then

echo No arguments given

exit 1

fi

if [ ! -f file-reports.info ]; then

touch file-reports.info

fi

if [ ! -f directory-reports.info ]; then

touch directory-reports.info

fi

echo "" > file-reports.info

echo "" > directory-reports.info

for var in $@; do

found=0

for F in `find .`; do

fname=`echo $F | awk -F/ '{print $NF}'`

if [ $var = $fname ]; then

if [ -f $F ]; then

found=1

size=`ls -l $F | awk '{print $5}'`

echo $F $size >> file-reports.info

elif [ -d $F ]; then

found=1

for file in `ls -S $F | sed "s/ /\n/gi"`; do

filepath="${F}/${file}"

size=`ls -l $filepath | awk '{print $5}'`

if [ -f $filepath ]; then

echo $file $size >> directory-reports.info

break

fi

done

fi

fi

done

if [ $found -eq 0 ]; then

if echo $var | grep -E -q "^[aeiouAEIOU]"; then

touch $var

else

mkdir $var

fi

fi

done